1

INTRODUCCIÓN

Sin su software, la computadora es básicamente un montón de metal inútil. Con su software, una computadora puede almacenar, procesar y recuperar información; exhibir documentos multimedia; realizar búsquedas en Internet; y realizar muchas otras actividades valiosas para justificar su existencia. El software de computadora puede dividirse a grandes rasgos en dos tipos: programas de sistema, que controlan la operación de la computadora misma, y programas de aplicación, que realizan las tareas reales que el usuario desea. El programa de sistema más fundamental es el **sistema operativo**, que controla todos los recursos de la computadora y establece la base sobre la que pueden escribirse los programas de aplicación.

Un sistema de computadora moderno consiste en uno o más procesadores, memoria principal (también conocida como RAM, memoria de acceso aleatorio), discos, impresoras, interfaces de red y otros dispositivos de entrada/salida. A todas luces, se trata de un sistema complejo. Escribir programas que sigan la pista a todos estos componentes y los usen correctamente, ya no digamos óptimamente, es una tarea en extremo difícil. Si todos los programadores tuvieran que ocuparse de cómo trabajan las unidades de disco, y de las docenas de cosas que pueden fallar al leer un bloque de disco, es poco probable que pudieran escribirse muchos programas.

Hace muchos años se hizo muy evidente que debía encontrarse alguna forma de proteger a los programadores de la complejidad del hardware. La solución que ha evolucionado gradualmente consiste en poner una capa de software encima del hardware solo, que se encargue de administrar todas las partes del sistema y presente al usuario una interfaz o **máquina virtual** que sea más

fácil de entender y programar. Esta capa de software es el sistema operativo, y constituye el tema de este libro.

La situación se muestra en la Fig. 1-1. En la parte inferior está el hardware que, en muchos casos, también se compone de dos o más capas. La capa más baja contiene los dispositivos físicos, que consisten en *chips* de circuitos integrados, alambres, fuentes de potencia, tubos de rayos catódicos y otros aparatos físicos similares. La forma en que éstos se construyen y sus principios de funcionamiento pertenecen al campo del ingeniero electricista.

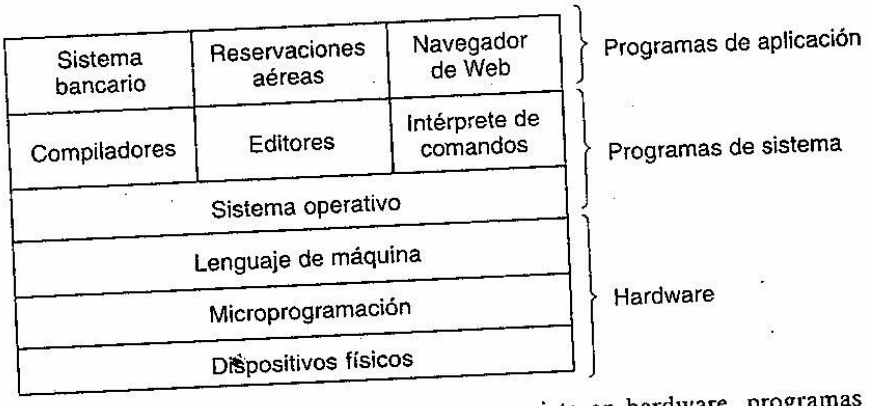

Figura 1-1. Un sistema de computadora consiste en hardware, programas de sistema y programas de aplicación.

A continuación (en algunas máquinas) viene una capa de software primitivo que controla directamente estos dispositivos y ofrece una interfaz más aseada a la siguiente capa. Este software, llamado **microprograma**, suele estar almacenado en memoria de sólo lectura. En realidad es un intérprete, que obtiene las instrucciones de lenguaje de máquina como ADD, MOVE y JUMP y las ejecuta en una serie de pasos pequeños. Por ejemplo, para ejecutar una instrucción ADD (sumar), el microprograma debe determinar dónde se encuentran los números que se van a sumar, obtenerlos, sumarlos y almacenar el resultado en algún lugar. El conjunto de instrucciones que el microprograma interpreta define el **lenguaje de máquina**, que no es realmente parte de la máquina física, aunque los fabricantes de computadoras siempre lo describen en sus manuales como tal, de modo que muchas personas piensan en él como si fuera la "máquina" real.

Algunas computadoras, llamadas **RISC (computadoras con conjunto de instrucciones reducido)**, no tienen un nivel de microprogramación. En estas máquinas, el hardware ejecuta las instrucciones de lenguaje de máquina directamente. Por ejemplo, el Motorola 680x0 tiene un nivel de microprogramación, pero el IBM PowerPC no.

El lenguaje de máquina por lo regular cuenta con entre 50 y 300 instrucciones, la mayor parte de ellas para trasladar datos dentro de la máquina, realizar operaciones aritméticas y comparar valores. En esta capa, los dispositivos de entrada/salida se controlan cargando valores en **registros de dispositivo** especiales. Por ejemplo, para un disco que ejecuta una lectura, sus registros se cargan con los valores de dirección del disco, dirección de memoria principal, conteo de bytes y dirección de acceso (READ o WRITE). En la práctica se requieren muchos parámetros más, y el

estado devuelto por la unidad de disco después de una operación es muy complejo. Además, la temporización desempeña un papel importante en la programación de muchos dispositivos de E/S.

Una función importante del sistema operativo es ocultar toda esta complejidad y ofrecer al programador un conjunto de instrucciones más cómodo con el que pueda trabajar. Por ejemplo, LEER BLOQUE DE ARCHIVO es conceptualmente más sencillo que tener que preocuparse por los detalles de mover cabezas de disco, esperar que se estabilicen, etcétera.

Encima del sistema operativo está el resto del software de sistema. Aquí encontramos el intérprete de comandos (shell), sistemas de ventanas, compiladores, editores y otros programas similares independientes de la aplicación. Es importante darse cuenta de que estos programas definitivamente no forman parte del sistema operativo, a pesar de que casi siempre son provistos por el fabricante de la computadora. Éste es un punto crucial, aunque sutil. El sistema operativo es la porción del software que se ejecuta en **modo kernel** o **modo supervisor**, y está protegido por el hardware contra la intervención del usuario (olvidándonos por el momento de algunos de los microprocesadores más viejos que no tienen ninguna protección de hardware). Los compiladores y editores se ejecutan en **modo de usuario**. Si a un usuario no le gusta un compilador en particular, él† está en libertad de escribir el suyo propio si lo desea; no está en libertad de escribir su propio manejador de interrupciones del disco, que forma parte del sistema operativo y normalmente está protegido por el hardware contra los intentos de los usuarios por modificarlo.

Por último, encima de los programas de sistema vienen los programas de aplicación. Los usuarios compran o escriben estos programas para resolver sus problemas particulares, como procesamiento de textos, hojas de cálculo, cálculos de ingeniería o juegos.

1.1 ¿QUÉ ES UN SISTEMA OPERATIVO?

La mayoría de los usuarios de computadora han tenido algo de experiencia con un sistema operativo, pero no es fácil precisar con exactitud qué es un sistema operativo. Parte del problema consiste en que el sistema operativo realiza dos funciones que básicamente no están relacionadas entre sí y, dependiendo de a quién le preguntemos, por lo general se nos habla principalmente de una función o de la otra. Veamos ahora las dos.

1.1.1 El sistema operativo como máquina extendida

Como ya dijimos, la **arquitectura** (conjunto de instrucciones, organización de memoria, E/S y estructura de *buses*) de la mayor parte de las computadoras en el nivel de lenguaje de máquina es primitiva y difícil de programar, sobre todo para entrada/salida. A fin de hacer más concreto este punto, veamos brevemente cómo se realiza la E/S de disco flexible usando el *chip* controlador NEC PD765 (o su equivalente), utilizado por la mayor parte de las computadoras personales. (En todo este libro usaremos indistintamente los términos "disco flexible" y "disquete".) El PD765

† "Él" debe leerse como "él o ella" a lo largo de todo el libro.

tiene 16 comandos, cada uno de los cuales se especifica cargando entre 1 y 9 bytes en un registro de dispositivo. Estos comandos sirven para leer y escribir datos, mover el brazo del disco y formatear pistas, así como para inicializar, detectar, restablecer y recalibrar el controlador y las unidades de disco.

Los comandos más básicos son READ y WRITE, cada uno de los cuales requiere 13 parámetros empacados en 9 bytes. Estos parámetros especifican cosas tales como la dirección del bloque de disco que se va a leer, el número de sectores por pista, el modo de grabación empleado en el medio físico, el espaciado de la brecha entre sectores y qué hacer con una marca de "dirección de datos eliminada". Si usted no entiende a qué nos referimos, no se preocupe; de eso se trata precisamente: es algo muy esotérico. Cuando se completa la operación, el *chip* controlador devuelve 23 campos de estado y error empacados en 7 bytes. Por si esto no fuera suficiente, el programador del disco flexible también debe tener presente en todo momento si el motor está encendido o apagado. Si el motor está apagado, debe encenderse (con un retardo de arranque largo) antes de que puedan leerse o escribirse datos. Empero, el motor no puede dejarse encendido demasiado tiempo, pues el disco flexible se desgastaría. Por tanto, el programador debe encontrar un equilibrio entre los retardos de arranque largos y el desgaste de los discos flexibles (y la pérdida de los datos que contienen).

Sin entrar en los *verdaderos* detalles, debe quedar claro que el programador ordinario seguramente no quiere intervenir de manera demasiado íntima en la programación de los discos flexibles (o de los duros, que son igualmente complejos, y muy distintos). En vez de ello, lo que el programador quiere es manejar una abstracción sencilla, de alto nivel. En el caso de los discos, una abstracción típica sería que el disco contiene una colección de archivos con nombre. Cada archivo puede abrirse para lectura o escritura, leerse o escribirse, y por último cerrarse. Los detalles de si la grabación debe usar o no modulación de frecuencia modificada y cuál es la situación actual del motor no deberán aparecer en la abstracción presentada al usuario.

El programa que oculta la verdad acerca del hardware y presenta al programador una vista sencilla y bonita de archivos con nombre que pueden leerse y escribirse es, por supuesto, el sistema operativo. Así como el sistema operativo aísla al programador del hardware del disco y presenta una interfaz sencilla orientada a archivos, también oculta muchos asuntos desagradables referentes a interrupciones, temporizadores, administración de memoria y otras funciones de bajo nivel. En cada caso, la abstracción que el sistema operativo ofrece es más sencilla y fácil de usar que el hardware subyacente.

En esta vista, la función del sistema operativo es presentar al usuario el equivalente de una **máquina extendida** o **máquina virtual** que es más fácil de programar que el hardware subyacente. La forma en que el sistema operativo logra este objetivo es una historia larga, que estudiaremos con detalle a lo largo del libro.

1.1.2 El sistema operativo como administrador de recursos

El concepto del sistema operativo como algo cuya función primordial es ofrecer a los usuarios una interfaz cómoda es una visión descendente. Una visión ascendente alternativa postula que el

sistema operativo está ahí para administrar todos los componentes de un sistema complejo. Las computadoras modernas constan de procesadores, memorias, temporizadores, discos, ratones, interfaces con redes, impresoras láser y una gran variedad de otros dispositivos. En la visión alternativa, la misión del sistema operativo es asegurar un reparto ordenado y controlado de los procesadores, memorias y dispositivos de E/S entre los diferentes programas que compiten por ellos.

Imagine lo que sucedería si tres programas que se ejecutan en alguna computadora trataran de imprimir sus salidas simultáneamente en la misma impresora. Las primeras líneas del listado podrían ser del programa 1, las siguientes del programa 2, luego algunas del programa 3, y así sucesivamente. El resultado sería un caos. El sistema operativo puede poner orden en el caos potencial almacenando temporalmente en el disco todas las salidas destinadas para la impresora. Cuando un programa haya terminado, el sistema operativo podrá copiar su salida del archivo de disco donde se almacenó a la impresora, mientras que el otro programa puede continuar generando salidas, ajeno al hecho de que dichas salidas no están yendo directamente a la impresora (todavía).

Cuando una computadora (o red) tiene múltiples usuarios, la necesidad de administrar y proteger la memoria, los dispositivos de E/S y demás recursos es aún mayor, ya que de otra manera los usuarios podrían interferirse. Además, es frecuente que los usuarios tengan que compartir no sólo hardware, sino también información (archivos, bases de datos, etc.). En pocas palabras, esta visión del sistema operativo sostiene que su tarea primordial es seguir la pista de quién está usando cuál recurso, atender solicitudes de recursos, contabilizar la utilización y mediar entre solicitudes en conflicto provenientes de diferentes programas y usuarios.

1.2 HISTORIA DE LOS SISTEMAS OPERATIVOS

Los sistemas operativos han estado evolucionando durante muchos años. En las siguientes secciones examinaremos brevemente este desarrollo. Dado que, históricamente, los sistemas operativos han estado de manera muy estrecha vinculados con la arquitectura de las computadoras en las que se ejecutan, estudiaremos las sucesivas generaciones de computadoras para ver qué clase de sistemas operativos usaban. Esta correspondencia entre las generaciones de sistemas operativos y de computadoras es algo burda, pero establece un poco de estructura que de otra forma sería inexistente.

La primera computadora digital verdadera fue diseñada por el matemático inglés Charles Babbage (1792-1871). Aunque Babbage invirtió la mayor parte de su vida y su fortuna tratando de construir su "máquina analítica", nunca logró que funcionara correctamente porque era totalmente mecánica, y la tecnología de su época no podía producir las ruedas, engranes y levas con la elevada precisión que él requería. Huelga decir que la máquina analítica no contaba con un sistema operativo.

Como acotación histórica interesante, diremos que Babbage se dio cuenta de que necesitaría software para su máquina analítica, así que contrató a una joven mujer, Ada Lovelace, hija del famoso poeta británico, Lord Byron, como la primera programadora de la historia. El lenguaje de programación Ada® recibió su nombre en honor a ella.